

Local, world-class
services for the
pharmaceutical industry


data management, data warehousing, statistics,
information technology and scientific writing

[Beyond Your Data]

Data analysis with R

Lecture 2

Useful functions

- In this lecture we look into some of the most commonly used R-functions.
 - These functions are linked to data manipulation, data derivation, creating new data, checking the qualities of the data and so on.
-  So all the things a data analyst always has to do, before actually getting into analyzing the data

cut()

- Cut-function divides the range of x into intervals and codes the values in x according to which interval they fall.
- So we can use it to create a factor from a numeric variable.

```
X <- 1:20
```

```
Xc <- cut(X,breaks=c(0,5,10,15,20),  
labels=c("Small","Medium","Large",  
"Huge"))
```

Conditional execution

- Conditionality in R is programmed with an if-statement.
- An if statement can be of the form *if (cond) expr* or of the form *if (cond) expr1 else expr2*.
 - *if(is.na(x)==F) {tf=1}*
 - *if(is.na(x)==F) {tf=1} else {tf=2}*
- NOTE! The if-statement does not affect the whole vector, only one element
 - Often if-statements are therefore used with eg. loops

For-loops

- For-loop is a common way to do looping in R, and works similarly as in other programming languages.
- Very simple example of a *for*-loop could be
 - `>for (i in 1:5) print(i)`
- Other looping constructs are *repeat()* and *while()*
- Both with *if* and *for* we can write *expression* without any brackets, if there is just one expression, otherwise we have to use curly brackets `{ }`.

Loops; problems

- Loops are not very efficient in R (nor in any other program). R has functions to avoid looping in some cases. For example apply-functions.
- However, loops are often the easiest way for a programmer to solve his/her problem without making mistakes.
 - The efficacy-problem only arises with bigger datasets, so in many cases we won't even notice a difference between loops and other solutions.

Apply-functions

- Apply-functions are a convenient way to apply function over an array
- Different apply-functions have been created for different situations
 - *apply()* applies function over array margins
 - We can calculate eg. sum of variables for each row of the data
 - *tapply()* applies a function to groups defined by factor
 - We can calculate eg. mean-concentrations separately for treatment-groups

Apply-functions

- With *sapply()* we can apply a function eg. to all columns of our data.
- Other apply-functions are:
 - *lapply()*
 - *mapply()*
 - *vapply()*
 - You can get familiar with these functions going through the help-pages.